1. Adding custom keys
2. maven project
3. Log4j2
4. git hub
5. java Terminology
6. JVM
7. java Features
8. data types(P& NP)
9. Sequence,Selection& Repetation
10. Class ,object,Method
11. Static
12. final
13. Pass by value & pass by refrance
14. types of members
15. Access modifires
16. Why only one public class in one file
17. Encapsulation
18. constructor(1. is it manditory  
     2. why prefered public)
19. This keyword
20. inheritance
21. Super keyword
22. Abstaction(Abstact class )
23. interface
24. Polymorphism 1)complie time (Overloading) 2)Runtime(Overriding)
25. Composition
26. Aggregation
27. Association
28. Generics
29. Junit
30. collection ans collections
31. linkedlist
32. stack
33. queue
34. hashmap
35. hashtable
36. binary search tree
37. regex
38. exception
39. lamada expresion
40. Stream API

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Adding custom keys**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1)go to windows

2)then go to pfreferance.

3)then go to java.

4)then go to Editor.

5)go to templets.

6)then click on New.

7)then add Name of shortcut,Discription & pattern.

8)click on applay.

9)click on Applay and Close.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Maven Project**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

maven is automation tool.it is used for manage dependacies in java.it handle required package for program exicution.

1)create maven project- click on file->then project-> then maven project.

2)Slect maven Artifact- filter with maven-archetype-quickstart-> then select org.apache.maven.archetype.

3)create maven project- enter -groupid:package prefix->Artifact id:project name->

package

pom.xml file is genrated.-

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. ... Other information such as the project version, description, developers, mailing lists and such can also be specified.

-in the pom.xml we need to add dependencies.

-add dependencies under

Dependencies

\*

\*

/ dependencies

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Log4j & Log4j2**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-It is used to printing or logging massage.

-if developer working on server side application where they cannot see whats going inside server ,then there only one visibility tool is a log file.

- if you print any through System.ou.println() then it routs to log file.

Log4j2 **–**it is a updated version of log4j library, it is simple, flexible, and fast.

**There are tree main component of log4j**

1)**Logger**- it is used to log the massage.

2)**Appende**r: it is used to publish the logging information to the destination like the file, database ,console etc.

3)**Layout**: layout is format of logging information in different style.

**Log4j2 with maven project**

1)Add log4j2 core and API dependencies in POM.xml (copy from google serch mvnrepository log4j2).

2)Add log4j2.xml to src/main/resources dictionary. (and add logxml code in there)

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<Configuration xmlns=*"http://logging.apache.org/log4j/2.0/config"*>

<Properties>

<Property name=*"basePath"*>logs</Property>

</Properties>

<Appenders>

<!-- File Appender -->

<File name=*"FILE"* fileName=*"${basePath}/logfile.log"* append=*"true"*>

<PatternLayout pattern=*"%-5p | %d{yyyy-MM-dd HH:mm:ss} | [%t] %C{2} (%F:%L) - %m%n"* />

</File>

<!-- Console Appender -->

<Console name=*"STDOUT"* target=*"SYSTEM\_OUT"*>

<PatternLayout pattern=*"%-5p | %d{yyyy-MM-dd HH:mm:ss} | [%t] %C{2} (%F:%L) - %m%n"* />

</Console>

</Appenders>

<Loggers>

<Logger name=*"com.jcg"* level=*"debug"* />

<Root level=*"info"*>

<AppenderRef ref=*"STDOUT"* />

<AppenderRef ref=*"FILE"* />

</Root>

</Loggers>

</Configuration>

create one program

:

**package** mavenproject.JunitDemo;

**import** org.apache.logging.log4j.LogManager;

**import** org.apache.logging.log4j.Logger;

/\*\*

\* Hello world!

\*

\*/

**public** **class** App

{

**private** **static** **final** Logger ***LOG*** =LogManager.*getLogger*(App.**class**);

**public** **static** **void** main( String[] args )

{

String massage="hello world";

***LOG***.debug(massage+"=will be perinted on debug");

***LOG***.info(massage+"will be perinted on info");

***LOG***.error(massage+"will be perinted on error");

***LOG***.warn(massage+"=will be perinted on warn");

***LOG***.info("Appending string: {}.",massage);

System.***out***.println( "Hello World!" );

}

}

-run this program.

- See the output in log4j2.log through terminal or git bash.

3)notice in log4j2.xml: base path-logging dir, Appenders-File & Console, Loggers-level & appender ref.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***JVM**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The JVM has two primary functions: to allow Java programs to run on any device or operating system (known as the "Write once, run anywhere" principle), and to manage and optimize program memory.

**Memory management in the JVM**

The most common interaction with a running JVM is to check the memory usage in the [heap and stack](https://www.javaworld.com/article/2077184/core-java/the-lean--mean--virtual-machine.html). The most common adjustment is [tuning the JVM's memory settings](https://www.javaworld.com/article/2078645/java-se/jvm-performance-optimization-part-3-garbage-collection.html).

**Garbage collection**

Before Java, all program memory was managed by the programmer. In Java, program memory is managed by the JVM. The JVM manages memory through a process called *garbage collection*, which continuously identifies and eliminates unused memory in Java programs. Garbage collection happens inside a running JVM.

When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Java Features\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Simple and Familiar

Compiled and Interpreted

Platform Independent

Portable

Architectural Neutral

Object-Oriented

Robust

Secure

Distributed

Multi-threaded and Interactive

High Performance

Dynamic and Extensible

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Access Modifiers\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Access modifiers** are object-oriented programming that is used to set the accessibility of classes, constructors, methods, and other members of Java.  
Using the access modifiers we can set the scope or accessibility of these classes, methods, constructors, and other members.

**Four Types of Access Modifiers**

* **Private**:We can access the **private modifier**only within the same class and not from outside the class.

* **Default:** We can access the **default modifier**only within the same package and not from outside the package. And also, if we do not specify any access modifier it will automatically consider it as **default**.
* **Protected**: We can access the **protected modifier** within the same package and also from outside the package with the help of the **child class.**If we do not make the child class, we cannot access it from outside the package. So **inheritance**is a must for accessing it from outside the package.
* **Public**: We can access the **public modifier** from anywhere. We can access **public modifiers** from within the class as well as from outside the class and also within the package and from outside the package.

\*\*\*\*\* Why only one public class in one file\*\*\*\*\*\*\*\*\*\*\*\*\*

* There can be only one public class in a java file because the **name of java file is same as the name of public class**.And obviously we can't have a file with two different names.
* Public class is always declare which has main method, when we declare two class public then compiler confused where is main method. That’s why it shows compile time error.

To resolve this either you need to shift one of the classes into a separate file or,

* Remove the public declaration before the class that doesn’t contain a *public static void main(String args)* method.
* Name the file with the class name that contains main method.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Java Array\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location



Types or Array->

\* Single Dimensional Array

\* Multidimensional Array

- Single Dimensional Array in Java

- Syntax to Declare an Array in Java

dataType[] arr; (or)

dataType []arr; (or)

dataType arr[];

-Instantiation of an Array in Java:

arrayRefVar=new datatype[size];

EX:

class Testarray{

public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation

a[0]=10;//initialization

a[1]=20;

a[2]=70;

a[3]=40;

a[4]=50;

//traversing array

for(int i=0;i<a.length;i++)//length is the property of array

System.out.println(a[i]);

}}

EX2:

int a[]={33,3,4,5};//declaration, instantiation and initialization

-Multidimensional Array in Java

In such case, data is stored in row and column based index (also known as matrix form).

declare Array:

dataType[][] arrayRefVar; (or)

dataType [][]arrayRefVar; (or)

dataType arrayRefVar[][]; (or)

dataType []arrayRefVar[];

Initialization of array:

int[][] arr=new int[3][3];//3 row and 3 column